UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/672,975	09/26/2003	Robin Alexis Takasugi	10014268-1	3620

22879 7590 04/14/2008

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
TSAI, SHENG JEN	
ART UNIT	**PAPER NUMBER**
2186	

NOTIFICATION DATE	DELIVERY MODE
04/14/2008	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

JERRY.SHORMA@HP.COM
mkraft@hp.com
ipa.mail@hp.com

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/672,975
Filing Date: September 26, 2003
Appellant(s): TAKASUGI ET AL.

Jeff A. Holmen
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 2/5/2007 appealing from the Office action mailed 8/9/2006.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Mater

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal Identifies the ground of rejections and the associated claims under rejection to be reviewed on appeal.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

6,092,149	Hicken et al.	7-18-2000

(9) Grounds of Rejection

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Claims 1-30 are rejected under 35 U.S.C. 102(b) as being anticipated by Hicken et al. (US 6,092,149).

As to claim 1, Hicken et al. disclose **a prefetch controller** [Disk Drive Cache System using a Dynamic priority Sequential Stream of Data Segments Continuously Adapted according to Prefetched Sequential, Random and Repeating Types of Accesses (title)] **for controlling retrieval of data from a data storage device** [the MEDIA storage, figure 1A, 40] **in response to a current host command received from a host device** [Host Computer, figure 1A, 50; figure 16B shows the flowchart in which a first and a second commands are received from the host computer], **the prefetch controller comprising**:

a sequential read detector [each segment is monitored to determine access types such as sequential, random, and repeating (abstract); figure 16B shows the flowchart of a sequential read detector in which a first and a second commands are received from the host computer, and a decision is made regarding if the first and the second commands constitute a sequential access based on the address of the required data (figure 16B, steps 1146, 1148 and 1150); when a sequential read stream is recognized, a different caching system path is used to process the commands in that

stream (column 11, lines 55-60)] **configured to generate a new sequential read indication for the current host command if the current host command and a previously received host command specify read operations that are non-sequential** [each segment is monitored to determine access types such as sequential, random, and repeating (abstract); figure 16B shows the flowchart of a sequential read detector in which a first and a second commands are received from the host computer, and a decision is made regarding if the first and the second commands constitute a sequential access based on the address of the required data (figure 16B, steps 1146, 1148 and 1150); step 1150 indicates that it is non-sequential]; **and a transfer length generator** [figure 1F shows that the transferred data comprises the "requested data" and the "prefetched data," and the transfer length value is the sum of the length of the requested data and the length of the prefetched data; For every read command the invention determines how much data to prefetch after the requested data is retrieved (column 10, lines 62-67)] **configured to provide a first transfer length value to the data storage device if the new sequential read indication is generated for the current host command, and provide a second transfer length value to the data storage device if the new sequential read indication is not generated for the current host command** [Each segment is monitored to determine access types such as sequential, random, and repeating. The access type determines the amount of data to prefetch and to save, including a minimum and maximum prefetch (abstract); When the adaptive prefetching is enabled and Mode Page 8, as described below, Min and Max Prefetch are 0xFFFF, the min and max prefetch are adaptively determined based

on the way data is accessing in this segment. If this cache access is a sequential stream, both min and max prefetch are be set to the number of blocks in a segment to fill a segment's worth of data. The system then discards requested data for this command once the data has been transferred. If this cache access is a repetitive type of access, the min is set to the blockCount of the command and the max is the number of blocks in a segment less the blockCount of the command in order to keep the requested data for possible repeated access in subsequent commands. The default values are zero for the min and blocks per segment for the max; this allows the prefetch to be interrupted as soon as a seek can be started for a new command, but fills the segment with new data if the prefetch is not interrupted (column 10, lines 62-67; column 11, lines 1-13); Figure 1G illustrates various types of accesses, including the sequential access (158), skip ahead access (166, 170 and 172), new (174 and 176), and repeated access (164 and 168); detailed description of each type of access is provided in column 9, lines 46-67 and column 10, lines 1-13. Note that only access type 158 is sequential and all the other types are non-sequential]; **and wherein the first transfer length value is determined by adding a prefetch value to a transfer length value specified in the current host command** [figure 1F shows that the transferred data comprises the "requested data" and the "prefetched data," and the transfer length value is the sum of the length of the requested data and the length of the prefetched data; figure 1C, step 126 performs "compute prefetch" to determine the length of prefetched data, followed by step 132, which "set buffer counter and start the disk" to set buffer counter to be the total length of data to be transferred and then start

the disk to transfer data to the disk; For every (current) read command the invention determines how much data to prefetch after the requested data (from the previous command) is retrieved (column 10, lines 62-67)].

As to claim 2, Hicken et al. teach that **the first transfer length value is larger than the second transfer length value** [The access type determines the amount of data to prefetch and to save, including a minimum and maximum prefetch (abstract); the value of the first transfer length and the second transfer length depends on the length of the "requested data" and the "prefetched data" as shown in figure 1F, and the transfer length value is the sum of the length of the requested data and the length of the prefetched data; assuming that the length of the requested data being the same for both the first and the second commands, the length of the first transfer and the second transfer would depend on the length of the prefetched data; the length of the prefetched data depends on the parameters of minimum and maximum prefetch values that are dynamically and adaptively determined as shown in figures 8E and 8B, which may result in that the first transfer length value being larger than the second transfer length value].

As to claim 3, Hicken et al. teach that **the sequential read detector comprises**: **operation compare logic configured to compare an operation specified in the current host command to an operation specified in the previously received host command, and generate a first indication for the current host command if the compared operations are both read operations** [command manager, figure 1B, 302; receive and decode command, figure 2A, 318; when a sequential read stream is

recognized, a different caching system path is used to process the commands in that stream … (column 11, lines 55-60); When sequential write commands are received … (column 11, lines 65-67)].

As to claim 4, Hicken et al. teach that **the sequential read detector further comprises**:

address compare logic configured to compare a first address associated with the current host command to a second address associated with the previously received host command, and generate a second indication for the current host command if the compared addresses are indicative of sequential operations [figure 16B shows the flowchart of a sequential read detector in which a first and a second commands are received from the host computer, and a decision is made regarding if the first and the second commands constitute a sequential access based on the address of the required data (figure 16B, step 1146); when a sequential read stream is recognized, a different caching system path is used to process the commands in that stream (column 11, lines 55-60)].

As to claim 5, Hicken et al. teach that **the sequential read detector further comprises: a sequential read indication generator configured to generate the new sequential read indication if the first and the second indications are not generated for the current host command** [figure 16B shows the flowchart of a sequential read detector in which a first and a second commands are received from the host computer, and a decision is made regarding if the first and the second commands constitute a sequential access based on the address of the required data (figure 16B,

step 1146); when a sequential read stream is recognized, a different caching system path is used to process the commands in that stream (column 11, lines 55-60)].

As to claim 6, Hicken et al. teach that **the sequential read detector comprises**: **a plurality of registers** [Tables 1,2 and 4 show a plurality of registers indicating the type of access; set up registers, figure 2C, step 222] **for storing an opcode specified in the current host command** [needs at least to determine whether this is a "read" or "write" operation], **an opcode specified in the previous host command** [figure 16B shows the flowchart of a sequential read detector in which a first and a second commands are received from the host computer, and a decision is made regarding if the first and the second commands constitute a sequential access based on the address of the required data (figure 16B, step 1146); when a sequential read stream is recognized, a different caching system path is used to process the commands in that stream (column 11, lines 55-60)], **a start address associated with the current host command** [the Logical Block Address (LBA), figure 1D, 180; "is the first logical block address of the data requested in the second command equal to the first logical block address of the data in the prefetch for the first command?", figure 16B, step 1146], **and an end address associated with the previous host command** [the Logical Block Address (LBA), figure 1D, 180; "is the first logical block address of the data requested in the second command equal to the first logical block address of the data in the prefetch for the first command?", figure 16B, step 1146].

As to claim 7, Hicken et al. teach that **the sequential read detector further comprises**:

opcode compare logic for comparing the stored opcodes [needs at least to determine whether this is a "read" or "write" operation];

address increment logic for incrementing the stored end address, thereby generating an incremented end address [address manipulation logics in place to support the determination of the various types of access shown in figure 1G including the sequential access (158), skip ahead access (166, 170 and 172), new (174 and 176), and repeated access (164 and 168), as the determination of access type is based on the beginning and ending addresses of the requested data compared to the beginning and ending addresses of the cached data as illustrated in figure 1G; figure 1D shows the LBA, OFFSET, PF LBA and END PF LBA address associated with the cache memory structure]; **and**

address compare logic for comparing the stored start address and the incremented end address [address manipulation logics in place to support the determination of the various types of access shown in figure 1G including the sequential access (158), skip ahead access (166, 170 and 172), new (174 and 176), and repeated access (164 and 168), as the determination of access type is based on the beginning and ending addresses of the requested data compared to the beginning and ending addresses of the cached data as illustrated in figure 1G; figure 1D shows the LBA, OFFSET, PF LBA and END PF LBA address associated with the cache memory structure].

As to claim 8, Hicken et al. teach that **the sequential read detector further comprises:**

a sequential read indication generator configured to generate the new sequential read indication based on outputs of the opcode compare logic and the address compare logic [figure 16B shows the flowchart of a sequential read detector in which a first and a second commands are received from the host computer, and a decision is made regarding if the first and the second commands constitute a sequential access based on the address of the required data (figure 16B, step 1146); when a sequential read stream is recognized, a different caching system path is used to process the commands in that stream (column 11, lines 55-60)].

As to claim 9, Hicken et al. teach that **the transfer length generator comprises:**

a first register for storing the prefetch value [the corresponding first register is the MAXPREFETCH register (figure 6C, steps 586 and 592), which is also shown in TABLE 1 (col. 7, lines 35-50) as CH_ENV_ADC_MAXPF; note that MINPREFETCH is set to BLOCKPERSEG in figure 6C, step 592; When the adaptive prefetching is enabled and Mode Page 8, as described below, Min and Max Prefetch are 0xFFFF, the min and max prefetch are adaptively determined based on the way data is accessing in this segment (column 10, lines 62-67; column 11, lines 1-13); For every read command the invention determines how much data to prefetch after the requested data is retrieved (column 10, lines 62-67); note that the min and max prefetch represents two registers specifying the prefetch values];

a second register for storing a zero value [the corresponding second register is the MINPREFETCH register (figure 6C, steps 586 and 592), which is also shown in TABLE

1 (col. 7, lines 35-50) as CH_ENV_ADC_MINPF; note that MINPREFETCH is set to 0 in figure 6C, steps 586 and 592; The default values are zero for the min and blocks per segment for the max; this allows the prefetch to be interrupted as soon as a seek can be started for a new command, but fills the segment with new data if the prefetch is not interrupted (column 11, lines 9-13)]; **and**

a multiplexer coupled to the first and the second registers [the corresponding multiplexing function is shown in figure 8D: COMPUTE PREFETCH, where either the value of the MINPREFETCH register (step 692) or the value of the MAXPREFETCH register (step 694) is selected, depending on the condition of step 690]**, the multiplexer responsive to the new sequential read indication for selectively outputting the prefetch value or the zero value** [the corresponding multiplexing function is shown in figure 8D: COMPUTE PREFETCH, where either the value of the MINPREFETCH register (step 692) or the value of the MAXPREFETCH register (step 694) is selected, depending on the condition of step 690; note that MINPREFETCH is set to BLOCKPERSEG in figure 6C, step 592 and that MINPREFETCH is set to 0 in figure 6C, steps 586 and 592; The default values are zero for the MIN and blocks per segment for the MAX; this allows the prefetch to be interrupted as soon as a seek can be started for a new command, but fills the segment with new data if the prefetch is not interrupted (column 10, lines 62-67; column 11, lines 1-13); Figure 1G illustrates various types of accesses, including the sequential access (158), skip ahead access (166, 170 and 172), new (174 and 176), and repeated access (164 and 168)].

As to claim 10, Hicken et al. teach that **the transfer length generator further comprises:**

a third register for storing the transfer length value specified in the current host command [the transferred data comprises the "requested data" and the "prefetched data" as shown in figure 1F, and the transfer length value is the sum of the length of the requested data and the length of the prefetched data; figure 1D shows the storage of the parameter "BLOCK COUNT"].

As to claim 11, Hicken et al. teach that **the transfer length generator further comprises**:

an adder for adding the value stored in the third register and the value output by the multiplexer the transferred data comprises the "requested data" and the "prefetched data" as shown in figure 1F, and the transfer length value is the sum of the length of the requested data and the length of the prefetched data; address manipulation logics in place to support the determination of the various types of access shown in figure 1G including the sequential access (158), skip ahead access (166, 170 and 172), new (174 and 176), and repeated access (164 and 168), as the determination of access type is based on the beginning and ending addresses of the requested data compared to the beginning and ending addresses of the cached data as illustrated in figure 1G; figure 1D shows the LBA, OFFSET, PF LBA and END PF LBA address associated with the cache memory structure; figure 1D shows the equation of "ADDRESS OF LBA = (ADDR OF SEG) + (OFFSET * SECTOR SIZE)"].

As to claim 12, refer to "As to claim 1" presented earlier in this Office Action.

As to claims 13-14, Hicken et al. teach **buffering the data** [figure 1E shows the buffer] **received from the storage device and outputting the buffered data to the host** [figure 1F shows how the requested data and the prefetch data is arranged in the buffer; figure 1G shows how data may be buffered depending on the access type].

As to claim 15, refer to "As to claim 3" and "As to claim 4."

As to claim 16, refer to "As to claim 9" and "As to claim 11."

As to claim 17, refer to "As to claim 1" presented earlier in this Office Action.

As to claim 18, refer to "As to claim 15."

As to claim 19, refer to "As to claim 9" and "As to claim 11."

As to claim 20, refer to "As to claim 1," "As to claim 12," and "As to claim 17."

As to claim 21, refer to "As to claim 2."

As to claim 22, refer to "As to claim 3."

As to claim 23, refer to "As to claim 4."

As to claim 24, refer to "As to claim 5."

As to claim 25, Hicken et al. teach that **the computer-readable medium of claim 20, wherein the method further comprises**:

storing an opcode specified in the current host command, an opcode specified in the previous host command, a start address associated with the current host command, and an end address associated with the previous host command [figure 16B shows the flowchart of a sequential read detector in which a first and a second commands are received from the host computer, and a decision is made regarding if the first and the second commands constitute a sequential access based

on the address of the required data (figure 16B, step 1146); when a sequential read stream is recognized, a different caching system path is used to process the commands in that stream (column 11, lines 55-60)].

As to claim 26, refer to "As to claim 9" and "As to claim 11."

As to claim 27, refer to "As to claim 9" and "As to claim 11."

As to claim 28, refer to "As to claim 9."

As to claim 29, refer to "As to claim 10."

As to claim 30, refer to "As to claim 11."

(10) Response to Arguments

Appellants' arguments have been fully and carefully considered with Examiner's answers set forth below.

Response to Arguments (A) on Rejections of Claims 1-8

Appellants contend that claims 1-8 are patentable over Hicken et al. (US 6,092,149, hereinafter referred to as Hicken) as, allegedly, Hicken fails to teach, or suggest "adding a prefetch value to a transfer length value specified in a current non-sequential read command, and then providing this sum to a data storage device." The Examiner disagrees with this argument for the following reasons:

First, figure 1F of Hicken shows that the transferred data comprises the "requested data" and the "prefetched data," and the transfer length value is the sum of the length of the requested data and the length of the prefetched data. This clearly suggests that the length of data to be transferred is the sum of the length of the "requested data" and the length of the "prefetched data."

Second, figure 1D of Hicken also shows the segment associated with each cache entry comprises a LBA (Logic Base Address, also shown in figure 1F, 180) and a PF LBA (PreFetch Logic Base Address, also shown in figure 1F, 182). Since the caching system divides the cache memory into segments to store multiple streams of data and each segment is transferred as a unit [abstract]. This provides another piece of evidence that that the length of data to be transferred, as a segment, is the sum of the length of the "requested data" and the length of the "prefetched data."

Third, figure 1C of Hicken further shows the flow chart of the overall process. Significantly, it illustrates step 126 performs "compute prefetch" to determine the length of prefetched data, followed by step 132, which "set buffer counter and start the disk" to set buffer counter to be the total length of data to be transferred and then start the disk to transfer data to the disk. This provides yet another piece of evidence that that the length of data to be transferred, as a segment, is the sum of the length of the "requested data" and the length of the "prefetched data."

Appellants further argue that the prefetch in Hicken is a separate transfer that is determined after the requested data is retrieved, thus Hicken fails to teach, or suggest "adding a prefetch value to a transfer length value specified in a current non-sequential read command, and then providing this sum to a data storage device." The Examiner disagrees with this argument for the following reasons:

First, Hicken's invention is directed toward dynamically prefetching data to maximize disk drive performance based on past access history [abstract]. Particularly, the amount of data to be prefetched for the current read command depends on the data

acquired during the previous read commands as well as the type of access [figure 1G illustrates several types of access, including "full cache hit," "partial cache hit," "skip ahead cache hit," "sequential cache hit" and "no cache hit;" column 9, lines 28-67].

For instance, if the data acquired in previous read commands (the prefetched data is stored in the cache buffer [figure 1A, 10]) already includes portion of the data that is requested by the current read command, then the amount of data to be prefetched for the current read command needs to be adjusted accordingly [the caching system may prescan the cache memory during prefetch to alter the prefetch amount is response to a command request (abstract); figures 8A~8E shows the details of adjusting the prefetch amount].

Second, the citation "For every read command the invention determines how much data to prefetch after the requested data is retrieved (column 10, lines 62-67)" means determining the amount of data to be prefetched for the current read command after the requested data of the previous read command is retrieved.

This is illustrated in more details in figures 10B~10F. In figure 10B, step 725 determines if this is a "read" command, if it is a "read" command, step B2 follows. In figure 10C, step B2, step 754 determines if this is a "partial hit in prefetch and prefetch from previous command will fetch a higher LBA than the current command." If the answer is "YES," the prefetch length is adjusted to accommodate data already requested, as stated in step 756.

Third, the requested data and prefetch data are retrieved from the disk drive [figure 1, 40] at the same time rather than separately. This is illustrated in figure 1C. In

figure 1C, Step 104 determines if there is a command from the host to be processed. Step 126 computes the prefetch length, followed by Step 132, which set buffer counter (to inform the disk and to monitor how much data to be transferred from the disk for this transfer operation) and start the disk (instruct the disk to begin transfer data). Note that in the entire flowchart only one step, Step 132, involves data transferring from the disk. Thus all data transfer from the disk to the cache buffer is performed only once, not twice for requested data and prefetch data separately.

Therefore, the Examiner's position regarding the patentability of claims 1-8 remains the same as indicated in the previous Office Action.

Response to Arguments (B) on Rejections of Claims 9-10

Appellants contend that claims 9-10 are allowable over Hicken simply by virtue of their inheriting the limitation of "adding a prefetch value to a transfer length value specified in a current non-sequential read command, and then providing this sum to a data storage device" as recited in independent claim 1. The Examiner disagrees.

The Examiner has fully addressed and demonstrated that Hicken indeed teach the limitation of "adding a prefetch value to a transfer length value specified in a current non-sequential read command, and then providing this sum to a data storage device" in the previous section of this Office Action. Refer to "***Response to Arguments (A) on Rejections of Claims 1-8***" for details.

Appellants also contend that, with respect to claim 9, the MIN and MAX prefetch values disclosed by Hicke are only "numbers" and not "registers." The Examiner disagrees.

TABLE 1 (col. 7, lines 35-50) of Hicken shows that the MAX prefetch variable is represented using a register designated as CH_ENV_ADC_MAXPF, and that the MIN prefetch variable is represented using a register designated as CH_ENV_ADC_MINPF. Thus, MAX and MIN are registers storing the values of the maximum and minimum prefetch value, respectively, and their values may be dynamically adjusted depending on the access type [abstract].

Appellants further contend that, with respect to claim 9, the Hicken reference fails to teach the element of multiplexer. The Examiner disagrees.

First, Hicken teaches that the corresponding multiplexing function is shown in figure 8D: COMPUTE PREFETCH, where either the value of the MINPREFETCH register (step 692) or the value of the MAXPREFETCH register (step 694) is selected, depending on the condition of step 690;

Second, Hicken shows that MINPREFETCH is set to BLOCKPERSEG in figure 6C, step 592 and that MINPREFETCH is set to 0 in figure 6C, steps 586 and 592. Thus the selection of either the value of the MINPREFETCH register (step 692) or the value of the MAXPREFETCH register (step 694) is to select a prefetch value that is either BLOCKPERSEG (when the MAXPREFETCH register is selected, step 694) or 0 (when the MINPREFETCH register is selected, step 692). Note also that a prefetch value of 0 simply means no prefetch.

Third, Hicken shows that the default values are zero for the MIN and blocks per segment for the MAX; this allows the prefetch to be interrupted (i.e., no prefetch) as

soon as a seek can be started for a new command, but fills the segment with new data if the prefetch is not interrupted (column 10, lines 62-67; column 11, lines 1-13).

Therefore, Hicken clearly teaches "multiplexing" the prefetch values of BLOCKPERSEG and 0 by using the MAXPREFETCH register and the MINPREFETCH register.

Therefore, the Examiner's position regarding the patentability of claims 9-10 remains the same as indicated in the previous Office Action.

Response to Arguments (C) on Rejections of Claim 11

Appellants contend that claim 11 is allowable over Hicken because the Examiner has not identified any structure in Hicken that corresponds to, or performs the functions of the adder recited in claim 11. The Examiner disagrees.

Claim 11 depends from claim 1, which recites the limitation of "adding a prefetch value to a transfer length value specified in a current non-sequential read command, and then providing this sum to a data storage device," and the Examiner has fully addressed and demonstrated that Hicken indeed teach this limitation of in the previous section of this Office Action. Refer to "***Response to Arguments (A) on Rejections of Claims 1-8***" for details.

It is inherent that "an adding function" has to be performed in order to "adding a prefetch value to a transfer length value specified in a current non-sequential read command, and then providing this sum to a data storage device," as recited in claim 1.

Therefore, the Examiner's position regarding the patentability of claim 11 remains the same as indicated in the previous Office Action.

Response to Arguments (D) on Rejections of Claims 12-16

Appellants contend that claims 12-16 are allowable over Hicken because, allegedly, Hicken fails to teach, or suggest "adding a prefetch value to a transfer length value specified in a current non-sequential read command, and then providing this sum to a data storage device." The Examiner disagrees.

The Examiner has fully addressed and demonstrated that Hicken indeed teach the limitation of "adding a prefetch value to a transfer length value specified in a current non-sequential read command, and then providing this sum to a data storage device" in the previous section of this Office Action. Refer to "***Response to Arguments (A) on Rejections of Claims 1-8***" for details.

Therefore, the Examiner's position regarding the patentability of claims 12-16 remains the same as indicated in the previous Office Action.

Response to Arguments (E) on Rejections of Claims 17-18

Appellants contend that claims 17-18 are allowable over Hicken because, allegedly, Hicken fails to teach, or suggest "adding a prefetch value to a transfer length value specified in a current non-sequential read command, and then providing this sum to a data storage device." The Examiner disagrees.

The Examiner has fully addressed and demonstrated that Hicken indeed teach the limitation of "adding a prefetch value to a transfer length value specified in a current non-sequential read command, and then providing this sum to a data storage device" in the previous section of this Office Action. Refer to "***Response to Arguments (A) on Rejections of Claims 1-8***" for details.

Therefore, the Examiner's position regarding the patentability of claims 17-18 remains the same as indicated in the previous Office Action.

Response to Arguments (F) on Rejections of Claim 19

Appellants contend that claim 19 is allowable over Hicken because the Examiner has not identified any structure in Hicken that corresponds to, or performs the functions of the adding means recited in claim 19. The Examiner disagrees.

Appellants contend that, with respect to claim 19, the MIN and MAX prefetch values disclosed by Hicke are only "numbers" and not "registers." The Examiner disagrees.

TABLE 1 (col. 7, lines 35-50) of Hicken shows that the MAX prefetch variable is represented using a register designated as CH_ENV_ADC_MAXPF, and that the MIN prefetch variable is represented using a register designated as CH_ENV_ADC_MINPF. Thus, MAX and MIN are registers storing the values of the maximum and minimum prefetch value, respectively, and their values may be dynamically adjusted depending on the access type [abstract].

Appellants further contend that, with respect to claim 19, the Hicken reference fails to teach the element of multiplexer. The Examiner disagrees.

First, Hicken teaches that the corresponding multiplexing function is shown in figure 8D: COMPUTE PREFETCH, where either the value of the MINPREFETCH register (step 692) or the value of the MAXPREFETCH register (step 694) is selected, depending on the condition of step 690;

Second, Hicken shows that MINPREFETCH is set to BLOCKPERSEG in figure 6C, step 592 and that MINPREFETCH is set to 0 in figure 6C, steps 586 and 592. Thus the selection of either the value of the MINPREFETCH register (step 692) or the value of the MAXPREFETCH register (step 694) is to select a prefetch value that is either BLOCKPERSEG (when the MAXPREFETCH register is selected, step 694) or 0 (when the MINPREFETCH register is selected, step 692). Note also that a prefetch value of 0 simply means no prefetch.

Third, Hicken shows that the default values are zero for the MIN and blocks per segment for the MAX; this allows the prefetch to be interrupted (i.e., no prefetch) as soon as a seek can be started for a new command, but fills the segment with new data if the prefetch is not interrupted (column 10, lines 62-67; column 11, lines 1-13).

Therefore, Hicken clearly teaches "multiplexing" the prefetch values of BLOCKPERSEG and 0 by using the MAXPREFETCH register and the MINPREFETCH register.

It is inherent that "an adding function" has to be performed in order to "adding a prefetch value to a transfer length value specified in a current non-sequential read command, and then providing this sum to a data storage device," as recited in claim 17.

Therefore, the Examiner's position regarding the patentability of claim 19 remains the same as indicated in the previous Office Action.

Response to Arguments (G) on Rejections of Claims 20-27

Appellants contend that claims 20-27 are allowable over Hicken because, allegedly, Hicken fails to teach, or suggest "adding a prefetch value to a transfer length

value specified in a current non-sequential read command, and then providing this sum to a data storage device." The Examiner disagrees.

The Examiner has fully addressed and demonstrated that Hicken indeed teach the limitation of "adding a prefetch value to a transfer length value specified in a current non-sequential read command, and then providing this sum to a data storage device" in the previous section of this Office Action. Refer to "***Response to Arguments (A) on Rejections of Claims 1-8***" for details.

Therefore, the Examiner's position regarding the patentability of claims 20-27 remains the same as indicated in the previous Office Action.

Response to Arguments (H) on Rejections of Claims 28-29

Appellants contend that claim 11 is allowable over Hicken because there is no teaching or suggestion in Hicken regarding selectively outputting a prefetch value or a zero value based on whether a new sequential read indication is generated for the current host command. The Examiner disagrees due to the following reasons:.

First, Hicken teaches that "each segment is monitored to determine **access types** such as sequential, random, and repeating. The **access type** determines the amount of data to prefetch and to save, including a minimum and maximum prefetch (abstract)."

Second, Hicken further teaches that "If this cache **access** is a sequential stream, both min and max prefetch are be set to the number of blocks in a segment to fill a segment's worth of data. The system then discards requested data for this command once the data has been transferred. If this cache **access** is a repetitive type of access,

the min is set to the blockCount of the command and the max is the number of blocks in a segment less the blockCount of the command in order to keep the requested data for possible repeated access in subsequent commands. **The default values are zero** for the min and blocks per segment for the max; this allows the prefetch to be interrupted as soon as a seek can be started for a new command, but fills the segment with new data if the prefetch is not interrupted (column 10, lines 62-67; column 11, lines 1-13)."

Therefore, the Examiner's position regarding the patentability of claim 11 remains the same as indicated in the previous Office Action.

Response to Arguments (I) on Rejections of Claim 30

Appellants contend that claim 30 is allowable over Hicken because, allegedly, Hicken fails to teach, or suggest "adding a prefetch value to a transfer length value specified in a current non-sequential read command, and then providing this sum to a data storage device." The Examiner disagrees.

The Examiner has fully addressed and demonstrated that Hicken indeed teach the limitation of "adding a prefetch value to a transfer length value specified in a current non-sequential read command, and then providing this sum to a data storage device" in the previous section of this Office Action. Refer to "***Response to Arguments (A) on Rejections of Claims 1-8***" for details.

Therefore, the Examiner's position regarding the patentability of claim 30 remains the same as indicated in the previous Office Action.

(11) Related Proceedings Appendix

None.

/Sheng-Jen Tsai/
Partial Signatory Examiner, Art Unit 2186

/Matt Kim/
Matthew Kim
Supervisory Patent Examiner
Art Unit 2186

/Manorama Padmanabhan/
Quality Assurance Specialist, TC2100, WG2180

March 25, 2008